DISTRIBUTED TARGET TRACKING BASED ON BELIEF PROPAGATION CONSENSUS

Vladimir Savic[*]*, Henk Wymeersch*[**]*, and Santiago Zazo*[*]

[*] Signal Processing Applications Group, Universidad Politecnica de Madrid
[**] Department of Signals and Systems, Chalmers University of Technology
emails: {vladimir,santiago}@gaps.ssr.upm.es, henkw@chalmers.se

ABSTRACT

Distributed target tracking in wireless sensor networks (WSN) is an important problem, in which agreement on the target state can be achieved using conventional consensus methods, which take long to converge. We propose distributed particle filtering based on belief propagation (DPF-BP) consensus, a fast method for target tracking. According to our simulations, DPF-BP provides better performance than DPF based on standard belief consensus (DPF-SBC) in terms of disagreement in the network. However, in terms of root-mean square error, it can outperform DPF-SBC only for a specific number of consensus iterations.

Index Terms— Consensus, belief propagation, distributed target tracking, particle filtering, wireless sensor networks

1. INTRODUCTION

Distributed target tracking in wireless sensor networks (WSN) is an important task for many applications in which a central unit is not available. For example, in emergency situations, such as fires or nuclear disasters, a WSN can be deployed to detect these phenomena. Once the phenomenon is detected, sensors start to sense their neighborhood and cooperatively track people and assets. As sensors may not survive deployment, it is important to achieve tracking in a manner that is fully asynchronous and robust to sensors failures, and in such a way that every sensor has the same knowledge of the target location. Moreover, due to nonlinear relationships and possible non-Gaussian uncertainties, particle filtering (PF) may be preferred [1], instead of traditional methods based on Kalman filtering (KF).

Many of the methods for PF-based distributed target tracking in WSN are based on the construction and maintenance of the communication path [2, 3]. In [2], low-power sensors pass the parameters of likelihood function to the high-power sensors, which are responsible to manage the low-power nodes. In [3], a set of uncorrelated sensor cliques is constructed, in which slave nodes have to transmit Gaussian mixture parameters to the master node of the clique. Master node performs the tracking, and forward estimates to another clique. These *routing-based* algorithms lack robustness to failures and are also not suitable for ad-hoc sensor networks. To address these problems, several authors have considered using average consensus algorithms [4–6]. The global posterior distribution is approximated in [4] with a Gaussian mixture, and consensus is applied over the local parameters to compute the global parameters. Similarly, [5] uses a Gaussian approximation instead of Gaussian mixture. Randomized gossip consensus was used in [6] for distributed target tracking. A common drawback of these state-of-the-art methods is the slow convergence.

In this paper, we propose a novel method for target tracking using distributed particle filtering (DPF) based on belief propagation (BP) consensus. We perform simulations to analyze the performance of DPF-BP method, and compare with DPF based on standard belief consensus (DPF-SBC). According to our results, DPF-BP provides better performance than DPF-SBC in terms of disagreement in the network, as well as absolute accuracy, provided a specific number of consensus iterations is used.

2. PROBLEM FORMULATION

We consider N_s sensors with two-dimensional (2D) positions z_n ($n = 1, 2, \ldots, N$) and one target with an unknown state x_t at time t. The state of the target is defined as $x_t = [x_{1,t}\ x_{2,t}\ \dot{x}_{1,t}\ \dot{x}_{2,t}]^T$, where $x_{1,t}$ and $x_{2,t}$ represent 2D position of the target, and $\dot{x}_{1,t}$ and $\dot{x}_{2,t}$ the 2D velocity of the target. The goal of the WSN is to estimate x_t at each (discrete) time t. We use the following state-space model:

$$x_{t+1} = Ax_t + Bu_t \tag{1}$$

$$y_{n,t} = g_n(x_t) + v_{n,t}, \tag{2}$$

V. Savic is supported by the FPU fellowship from Spanish Ministry of Science and Innovation; This work is supported, in part, by the Swedish Research Council (VR), under grant No. 2010-5889; the European Research Council, under grant COOPNET No. 258418; the Spanish Ministry of Science and Innovation under the grants TEC2009-14219-C03-01 and TEC2010-21217-C02-02-CR4HFDVL; program CONSOLIDER-INGENIO 2010 under the grant CSD2008-00010 COMONSENS; the European Commission under the grant FP7-ICT-2009-4-248894-WHERE-2.

where $u_t = [u_{1,t}\, u_{2,t}]^T$ is the process noise due to the variation of the speed, $y_{n,t}$ is the local observation of sensor n at time t, and $v_{n,t}$ is its observation noise. The matrices A and B are given by

$$A = \begin{bmatrix} \boldsymbol{I}_2 & T_s\boldsymbol{I}_2 \\ \boldsymbol{0}_2 & \boldsymbol{I}_2 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{T_s^2}{2}\boldsymbol{I}_2 \\ T_s\boldsymbol{I}_2 \end{bmatrix}, \tag{3}$$

where T_s is the sampling interval, and $\boldsymbol{I}_2$ and $\boldsymbol{0}_2$ represent the identity and zero 2 x 2 matrices, respectively. We denote by G_t the set of the nodes that have a measurement available at time t. For the sake of concreteness, we assume that the measurements are distance measurements to the target, i.e., for $n \in G_t$, $g_n(x_t) = \left\| z_n - [x_{1,t}\ x_{2,t}]^T \right\|$. The measurement noise $v_{n,t}$ is distributed according to $p_v(\cdot)$, which may depend on measurement technique (e.g., acoustic, RSS) and the environment. The process noise u_t is distributed according to $p_u(\cdot)$. Finally, the sensors have an a priori distribution on their position $p_z(\cdot)$.

For simplicity, we assume ideal probability of detection for both sensing and communication range. That means that a sensor can detect the target if the distance between them is less than predefined value r, and that two sensors can communicate with each other if the distance between them is less than R.

3. CENTRALIZED TARGET TRACKING

We apply the Bayesian approach for this tracking problem and recursively determine the posterior distribution $p(x_t|y_{1:t})$ given the prior $p(x_{t-1}|y_{1:t-1})$, dynamic model $p(x_t|x_{t-1})$ defined by (1), and the likelihood function $p(y_t|x_t)$ defined by (2). We assume that $p(x_0|y_0) = p(x_0)$ is initially available. The posterior can be found using the prediction and filtering equations [1]:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{4}$$

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}). \tag{5}$$

Assuming independence among all measurements at time t, the global likelihood function $p(y_t|x_t)$ can be written as the product of the local likelihoods:

$$p(y_t|x_t) \propto \prod_{n \in G_t} p(y_{n,t}|x_t). \tag{6}$$

Note that $p(y_{n,t}|x_t)$ involves integrating out the position of the n-th sensor.

Due to the non-linear measurement model, we apply the particle filter (PF) [1], in which the posterior distribution is represented by a set of samples (particles) with associated weights. In order to avoid degeneracy problems (i.e., the situation in which all but one particle have negligible weights), we apply the *sample-importance-resampling* (SIR) method, in which the particles are drawn from $p(x_t|x_{t-1})$, then weighted by the likelihood function, $p(y_t|x_t)$, and finally resampled. We will refer to PF with SIR as centralized PF (CPF). CPF is run on one of the nodes in the WSN, which serves as fusion center. The main drawbacks of the CPF are: i) large energy consumption on the nodes which are in proximity of the fusion center, ii) high communication cost in large-scale networks; iii) the posterior distribution cannot be accessed from any node in the network; and iv) fusion center has to know the locations, observations, and observation models of all the nodes. In the following section, we will focus on distributed implementations of PF method, which alleviate these problems.

4. DISTRIBUTED TARGET TRACKING

Our goal is to track the target in a distributed, asynchronous way, such that all the nodes have a common view of the state of the target.

4.1. Distributed Particle Filtering

For a distributed implementation of the PF, we want to avoid exchanging measurements and to have a common set of samples and weights at every time step. If we can guarantee that the samples at time $t-1$ are common, and the weights at time t are common, then common samples at time t can be achieved by providing all nodes with the same seed for random number generation, so as to ensure that their pseudo-random generators are in the same state at all times. Ensuring common weights for all nodes can be achieved by means of a belief consensus (BC) algorithm. BC formally aims to compute, in a distributed fashion the product of a number of functions over the same variable

$$\mathrm{BC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \prod_{n=1}^{N_s} f_n(x). \tag{7}$$

However, most BC algorithms are not capable to achieve exact consensus in a finite number of iterations. As we require exact consensus on the weights, we additionally apply max-consensus [6, 7],

$$\mathrm{MC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \max_n f_n(x), \tag{8}$$

which computes the *exact* maximum over all arguments using the same asynchronous protocol as average consensus in a number of iterations (equal to the diameter of the graph, which represents the maximum hop distance between any two nodes). This idea has been already used in [6] for gossip-based consensus. The final algorithm is shown in Algorithm 1.

In the next sections, after description of the standard BC (SBC) algorithm, we propose novel BP consensus algorithm.

Algorithm 1 Distributed PF (DPF) (at node *n*, at time *t*)

1: **for all** particles $m = 1 : N_p$ **do**
2: Draw particle: $x_t^{(m)} \sim p(x_{n,t}|x_{t-1}^{(m)})$
3: Compute weight: $w_{n,t}^{(m)} = \mathrm{BC}\left(p(y_{1,t}|x_t^{(m)}), ..., p(y_{N_s,t}|x_t^{(m)})\right)$
4: **end for**
5: Normalize: $w_{n,t}^{(m)} = w_{n,t}^{(m)} / \sum_{m'} w_{n,t}^{(m')}$ (for $m = 1 : N_p$)
6: Compute estimates: $\hat{x}_{n,t} = \sum_m w_{n,t}^{(m)} x_{n,t}^{(m)}$
7: $\hat{w}_t^{(m)} = \mathrm{MC}\left(w_{1,t}^{(m)}, ..., w_{N_s,t}^{(m)}\right)$ (for $m = 1 : N_p$)
8: Normalize: $\hat{w}_t^{(m)} = \hat{w}_t^{(m)} / \sum_{m'} \hat{w}_t^{(m')}$ (for $m = 1 : N_p$)
9: Resample particles $\{x_t^{(m)}\}_{m=1}^{N_p}$ with replacement from $\{\hat{w}_t^{(m)}, x_t^{(m)}\}_{m=1}^{N_p}$

4.2. Standard Belief Consensus

Standard BC (SBC) [8] is defined in following iterative form:

$$M_n^{(i)}(x_t) = M_n^{(i-1)}(x_t) \prod_{u \in N_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-1)}(x_t)} \right)^{\epsilon}, \quad (9)$$

where N_n is the set of neighbors of node n, $M_n^{(i)}$ represents current estimate (at iteration i) of the global likelihood of the variable x_t (in our case, $x_t \in \{x_t^{(1)}, \ldots, x_t^{(N_p)}\}$), and $0 < \epsilon < 1/\eta_{\max}$, where $\eta_{\max}$ is maximum node degree in the network. We initialize by $M_n^{(1)}(x_t) = p(y_{n,t}|x_t)$, in which we assumed that $p(y_{n,t}|x_t) = 1$ for $n \notin G_t$. This consensus algorithm *guarantees* convergence (in all connected graphs) as the number of iterations goes to infinity [8]. Thus, it asymptotically converges to the geometrical average of the local distributions:

$$\lim_{i \to \infty} M_n^{(i)}(x_t) = \left(\prod_{n \in G_t} p(y_{n,t}|x_t) \right)^{1/N_s}, \quad (10)$$

from which the desired quantity, $\prod_{n \in G_t} p(y_{n,t}|x_t)$, can easily be found, for any value of $x_t \in \{x_t^{(1)}, \ldots, x_t^{(N_p)}\}$.

Note that the maximum node degree ($\eta_{\max}$) and number of nodes (N_s) must be known or estimated at every node.

4.3. Belief Propagation Consensus

Belief propagation (BP) [9,10] is a well-known message passing algorithm on an undirected graphical model. Motivated by its scalability, asynchronous behavior and robustness to failures, we apply it for consensus application. Consider the following function

$$\prod_n p(y_{n,t}|x_{n,t}) \prod_{u \in N_n} \delta(x_{n,t} - x_{u,t}), \quad (11)$$

which is equal to $\prod_{n \in G_t} p(y_{n,t}|x_t)$, whenever all the dummy variables $x_{u,t}$ take on the same value. Running BP on the corresponding graphical model yields the marginals $M_n(x_{n,t}) = C \prod_n p(y_{n,t}|x_{n,t})$ for every n, where C is a normalization constant. Note that this normalization constant is irrelevant, due to the normalization in in Algorithm 1, line 5. The BP message passing equations are now as follows: the belief at iteration i (the current approximation of $C \prod_n p(y_{n,t}|x_{n,t})$) is given by [11, eq. (8)]

$$M_n^{(i)}(x_{n,t}) \propto p(y_{n,t}|x_{n,t}) \prod_{u \in N_n} m_{un}^{(i)}(x_{n,t}), \quad (12)$$

while the message from node $u \in N_n$ to node n is given by [11, eq. (9)]

$$\begin{aligned} m_{un}^{(i)}(x_{n,t}) &\propto \int_{x_{u,t}} \delta(x_{n,t} - x_{u,t}) \frac{M_u^{(i-1)}(x_{u,t})}{m_{nu}^{(i-1)}(x_{u,t})} dx_{u,t} \\ &= \frac{M_u^{(i-1)}(x_{n,t})}{m_{nu}^{(i-1)}(x_{n,t})}. \end{aligned} \quad (13)$$

We note that since all dummy variables are the same, we can write $x_{n,t} = x_{u,t} = x_t$. Some straightforward manipulation yields

$$M_n^{(i)}(x_t) \propto M_n^{(i-2)}(x_t) \prod_{u \in N_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-2)}(x_t)} \right), \quad (14)$$

which represents novel BC algorithm based on BP. This method is initialized by $M_n^{(1)}(x_t) = p(y_{n,t}|x_t)$. We also need to set $M_n^{(2)}(x_t)$ in order to run the algorithm defined by (14). Using (12) and (13), and assuming that $m_{nu}^{(1)}(x_t) = 1$, we find

$$M_n^{(2)}(x_t) = p(y_{n,t}|x_t) \prod_{u \in N_n} p(y_{u,t}|x_t) \quad (15)$$

This algorithm guarantees convergence to $C \prod_n p(y_{n,t}|x_t)$ for cycle-free network graphs [10, 12]. When the network graph has cycles, the beliefs are only approximations of the true marginals. Comparing (14) and (9), we can see that, in contrast to SBC, BP-consensus agrees on product of all local evidences (not the N_s-th root of the product), and does not rely on knowledge of $\eta_{\max}$ and N_s. We refer to this variant of DPF as DPF-BP.

5. SIMULATION RESULTS

We will compare CPF, DPF-SBC, and DPF-BP methods, for a scenario with $N_s = 16$ sensors semi-randomly deployed in a 100 m $\times$ 100 m area (see Figure 2). The positions of these sensors have an a priori circular Gaussian distribution centered around the true position and with standard deviation

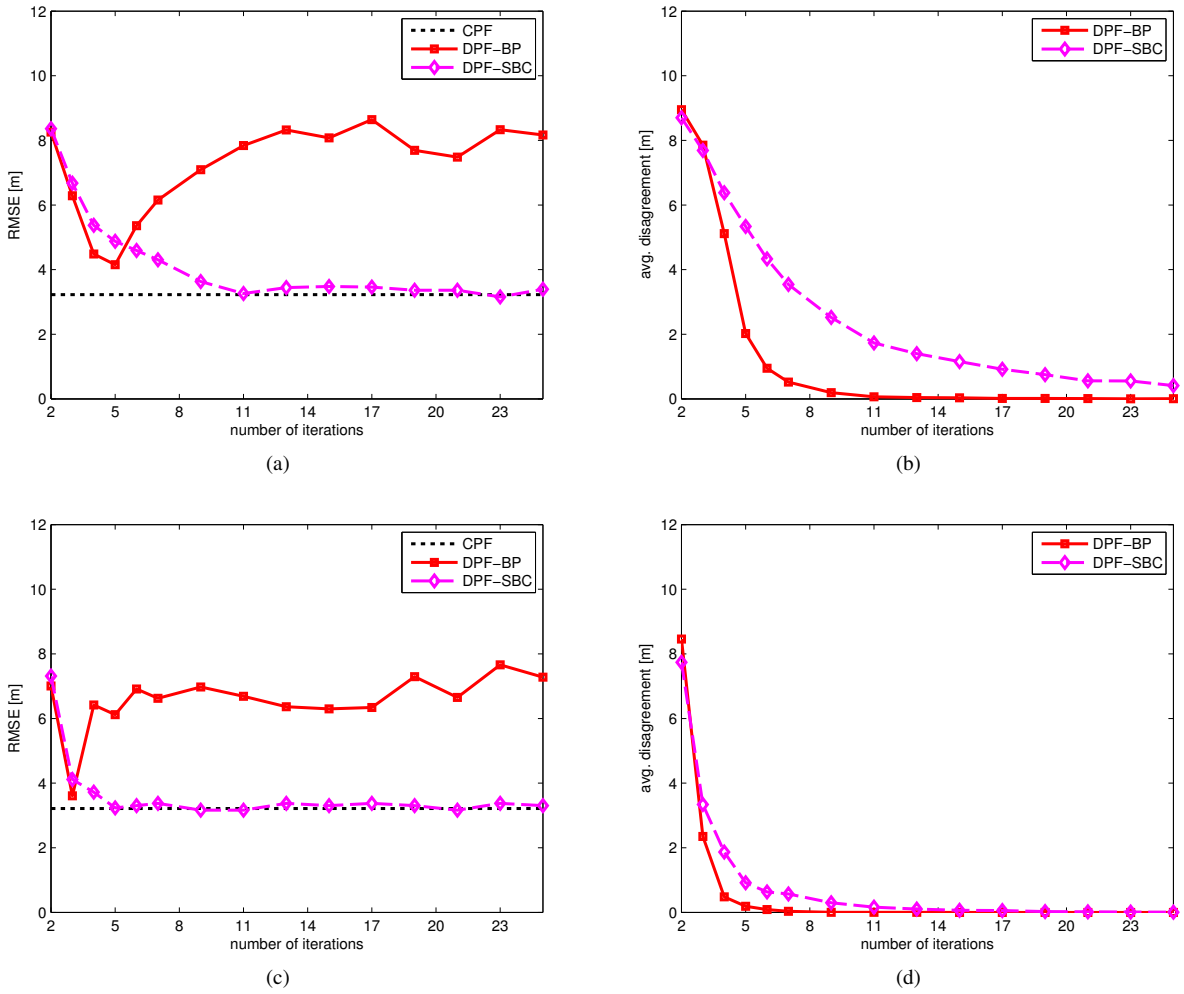

Fig. 1. Performance comparison of DPF methods as a function of the number of iterations. (a) RMSE, $R = 35$m, (b) avg. disagreement, $R = 35$m, (c) RMSE, $R = 55$m, and (d) avg. disagreement, $R = 55$m.

0.5 m in every direction. The target is moving with constant speed of 5 m/s according to a Gaussian random walk, during 40 time slots, each lasting $T_s = 1$ s. We set the sensing radius to $r = 30$ m, and consider two values of communication radius ($R = 35$ m and $R = 55$ m). We assume that the measured distance is distributed according to Gaussian mixture with two components.[1] We use $N_p = 400$ particles. For SBC, we used $\epsilon = 1/\eta_{\max}$ since it provides the fastest convergence [8]. The results are averaged over 200 Monte Carlo runs. We consider two performance metrics: root-mean-square error (RMSE) in the position error, and, for DPF methods, the average disagreement in the position error, defined as the difference between maximum and minimum error over network.

We will first investigate the convergence as a function of the number of iterations, for $R = 35$ m and $R = 55$ m (see Figure 1). We draw a number of conclusions. First of all, CPF provides the best RMSE performance, as it has access to all observations. DPF-SBC provides better RMSE performance than DPF-BP, as the latter algorithm is affected by the loops in the graph, leading to biased beliefs. However, we note that for a specific number of iterations (5 iter. for $R = 35$ m and 3 iter. for $R = 55$ m), DPF-BP outperforms DPF-SBC. In fact, using (14), it is possible to show that after $N_{\text{it}} = D_g + 1$ iterations (where D_g is diameter of the graph), all local likelihoods are available at each node, while during further iterations local likelihoods will be over-counted. Hence, at $N_{\text{it}} \approx D_g + 1$ DPF-BP achieves a good performance/delay trade-off, while DPF-SBC reaches guaranteed convergence in both metrics, but with a much longer delay.

In addition, we analyze DPF-SBC and DPF-BP at different time instants for a single run. We set $R = 35$ m, with

[1] A main component $\mathcal{N}(0\text{m}, (0.5\text{m})^2)$ with probability 0.9 and an outlier component $\mathcal{N}(5\text{m}, (0.5\text{m})^2)$ with probability 0.1.

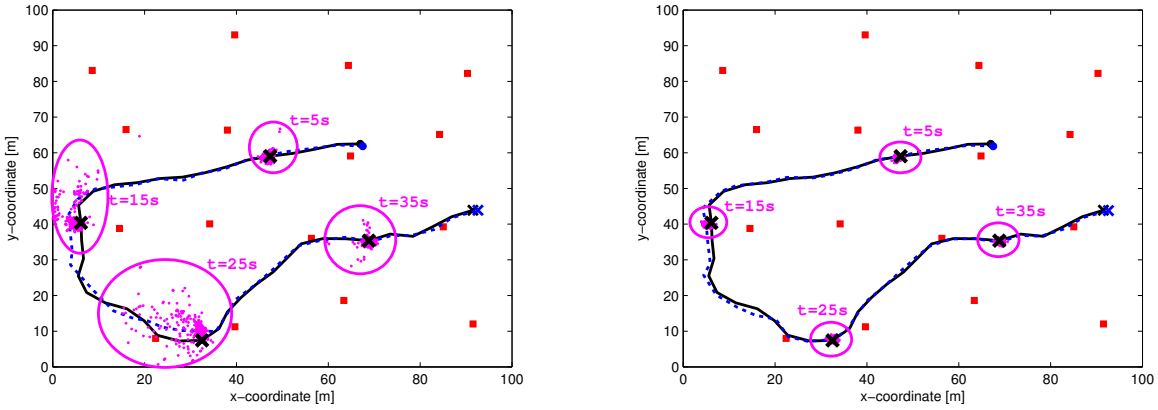

Fig. 2. Illustration of the particle clouds for: (a) DPF-SBC, and (b) DPF-BP. Sensors are marked with red squares, the true track with black line, estimated track with dashed line, and true positions of the targets (at 4 different time instants) with X.

$D_g = 4$ and fix $N_{\text{it}} = D_g + 1 = 5$. Therefore, it corresponds to scenario in which DPF-BP performs the best. According to Figure 2, we can see that the particles in DPF-BP are very informative (smallest spread), while particles in DPF-SBC are significantly more spread out. This is expected since we used max-consensus before resampling in order to ensure the same set of the particles. Since DPF-SBC converges significantly slower than DPF-BP (see Figure 1), every node may have a significantly different set of weights. Therefore, max-consensus over the weights will increase the amount of uncertainty. Moreover, both methods are fairly robust, since for all the methods the true position of the target is always within the particle cloud. This is important since these particles will be used for prediction and filtering in the next time instant (see Algorithm 1). Finally, note that if we increase the number of iterations, DPF-BP may provide a biased set of particles[2] (with the same spread), while DPF-SBC will become more accurate and informative.

6. CONCLUSION

We have proposed DPF-BP, a novel method for distributed target tracking. Compared to DPF-SBC, it provides significantly faster agreement of the estimates in the networks. It can also outperform DPF-SBC in terms of RMSE if we choose the right number of iterations. Future work includes online methods for estimating the optimal number of iterations, and the combination of SBC and BP, which might provide better a convergence/performance trade-off.

7. REFERENCES

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[2] M. Coates, "Distributed particle filters for sensor networks," in *Proc. of 3rd Workshop on Information Processing in Sensor Networks (IPSN)*, April 2004, pp. 99–107.

[3] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proc. of Fourth Int. Symp. Information Processing in Sensor Networks (IPSN)*, 2005, pp. 181–188.

[4] D. Gu, "Distributed particle filter for target tracking," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, April 2007, pp. 3856–3861.

[5] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Distributed Gaussian particle filtering using likelihood consensus," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3756–3759.

[6] D. Ustebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3296–3299.

[7] R. Olfati-Saber and R.M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520 – 1533, Sept. 2004.

[8] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," in *Proc. of NESC05 Worskhop*. 2006, pp. 169–182, Springer Verlag.

[9] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Uncertainty in Artificial Intelligence*, 2003, pp. 159–166.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988.

[11] A. T. Ihler, J. W. III Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, April 2005.

[12] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[2]The bias depends on the network configuration around the target.